GSAF: Efficient and Flexible Geocasting for Opportunistic Networks

Aydin Rajaei, Dan Chalmers, Ian Wakeman and George Parisis
School of Engineering and Informatics
University of Sussex, United Kingdom
Email: {a.rajaei, d.chalmers, ianw, g.parisis}@sussex.ac.uk

***Abstract*—With the proliferation of smartphones and their advanced connectivity capabilities, opportunistic networks have gained a lot of traction during the past years; they are suitable for increasing network capacity and sharing ephemeral, localised content. They can also offload traffic from cellular networks to device-to-device ones, when cellular networks are heavily stressed. Opportunistic networks can play a crucial role in communication scenarios where the network infrastructure is inaccessible due to natural disasters, large-scale terrorist attacks or government censorship. Geocasting, where messages are destined to specific locations (casts) instead of explicitly identified devices, has a large potential in real world opportunistic networks, however it has attracted little attention in the context of opportunistic networking.**

In this paper we propose *Geocasting Spray And Flood (GSAF)*, a simple but efficient and flexible geocasting protocol for opportunistic, delay-tolerant networks. GSAF follows a simple but elegant and flexible approach where messages take random walks towards the destination cast. Messages that follow directions away from the cast are extinct when the device buffer gets full, freeing space for new messages to be delivered. In GSAF, casts do not have to be pre-defined; instead users can route messages to arbitrarily defined casts. Our extensive evaluation shows that GSAF is efficient, in terms of message delivery ratio and latency as well as network overhead.

I. Introduction

The proliferation of smartphones and their long- and short-range connectivity capabilities, have made the deployment of opportunistic, delay-tolerant networks (DTNs) [1][2] reality [3][1]. Wireless technologies, such as LTE, Wi-Fi, Wi-Fi Direct and Bluetooth, allow smartphones to access the Internet as well as communicate with devices within their range, in an ad-hoc, peer-to-peer fashion [3][4]. Opportunistic networks have gained a lot of traction during the past years; they are suitable for increasing network capacity [5][6] and sharing ephemeral, localised content [7]. They are also appropriate for offloading traffic from cellular networks to device-to-device ones, whose formation is assisted by cellular providers [8][9], who have strong incentives to do so when their networks are heavily stressed [10][11]. Equally importantly, opportunistic networks can play a crucial role in communication scenarios where the network infrastructure is (partially or fully) inaccessible due to natural disasters, large-scale terrorist attacks or government censorship. They can also be the means for (localised) communication when the network infrastructure is not trusted. For example, FireChat[2]has been extensively used during the recent protests in Hong Kong (500,000 downloaded the application in Hong Kong alone during the first two weeks of the protests).

In most of the scenarios described above, communication and content dissemination is geographically confined (e.g. within a city or a region where a natural disaster took place or a part of the city where protesters demonstrate). Apart from being able to send messages to a specific device in the network (unicasting), it is also crucial to be able to route messages to specific geographical locations (*geocasting*) within the opportunistic network. Effective geocasting has a large potential in the real world use of opportunistic network: (1) geographical notification for emergency situations, such as fire and natural disasters; (2) location targeted advertising where a large volume of users is concentrated at specific locations (e.g. open festival venues or large stadiums) to attend music festivals, sports events or to participate in a demonstration; (3) geographically restricted service discovery. These geographical locations (*casts*) may be pre-defined, even before a network is deployed, or specified by the sender for each message, separately. A temporal aspect is also relevant to geocasting, apart from the spatial one; destination nodes must receive a message before it expires; e.g. before a notification becomes invalid in a natural disaster scenario.

Unicasting has been extensively studied in the context of DTNs [12], but none of the existing protocols can support geocasting, given that unicast protocols route messages to specific devices, which are explicitly identified by unique endpoint identifiers. However, little attention has been paid to geocasting, which has mostly been studied in the context of Mobile Ad Hoc Networks (MANETs) [13]. MANETs present radically different properties compared to opportunistic networks. In MANETs, connectivity (as well as the overall network topology) among mobile nodes is rather stable; no such assumptions can be made for DTNs, where mobility is high[3] and connectivity very intermittent. As a result, no end-to-end paths among all nodes exist at all times and no node knows the network topology, which constantly changes. Hence, none of the existing geocasting protocols for MANETs are suitable for opportunistic networks.

[1]http://tribehive.co.uk/

[2]http://tinyurl.com/ogsz75o

[3]Mobility is actually exploited so that messages are physically carried in the devices towards their final destinations.

In this paper we present *Geocasting Spray And Flood (GSAF)*, a simple but efficient and flexible geocasting protocol for opportunistic, delay-tolerant networks, which overcomes limitations of existing approaches. Contrary to protocols where casts must be pre-defined [14][15] or are defined as circles (by defining a centre point and a radius) [16], in our approach we allow for flexible definition of arbitrary casts based on a set of coordinates. The sender defines the cast, the cast definition is carried in the routed message and other nodes only check whether they reside within the defined cast. This flexibility is required by many communication scenarios where fine-grained specification of destination casts is crucial (e.g. fine-grained emergency notifications to avoid widespread panic). Moreover, in our approach a device can send a message in a cast even if it does not reside in it. This is in contrast to [7], where devices can only publish content within the region they reside. With such an approach, it would be very inefficient to reach relatively remote regions by just increasing the radius of the cast, effectively flooding a very large area with, probably, unwanted content.

In our approach, devices do not exchange any location-related information, thus preserving users' privacy, and take routing decisions autonomously. This is in contrast to approaches that exchange explicit [14] or aggregated location information (e.g. cast visiting probabilities [16]) or information that is used to collaboratively build mobility maps [15]. Exchanging location-relevant information consumes bandwidth and battery, resources that are precious, and rather scarce in opportunistic networks. Expensive computations (e.g. as in [15][16]) also drain the battery quickly. In [14] the network is partitioned into two layers, requiring either a third party to perform the partitioning or a distributed consensus protocol for electing nodes to be in each of these layers (consuming bandwidth especially under high node churn).

GSAF follows a simple but elegant approach where messages take random walks towards the destination cast. Messages that follow directions away from the cast are extinct when the device buffer gets full, freeing space for new messages to be delivered. In brief, message dissemination is as follows: a node receiving a message only has to check whether it is a destination node (i.e. it resides within the cast defined in the message) or not. This requires devices' location services and presents a well-known trade-off with respect to the accuracy of the reported location (which, in turn, affects the granularity of cast definition) and battery consumption[4]. In the latter case, a device carries and forwards the message to other nodes based on a ticketing mechanism, as in [17]. When a message reaches a cast, it is disseminated through controlled flooding and can never exit the borders of the destination cast. Expired messages are immediately deleted.

[4]In practice, in urban areas, a location accuracy of 20 - 50 meters, which is fine for many geocasting scenarios, can be achieved without GPS. If the network infrastructure is inaccessible, then one has to rely on GPS or collaborative localisation approaches, but our approach makes very light use of location services.

In Section IV we present an extensive evaluation of the proposed geocasting protocol. The results are very promising, indicating that our approach is viable and performs significantly better, in terms of message delivery ratio and latency as well as network overhead, compared to epidemic geocasting. Note that other approaches that employ computations of visiting probabilities for each message (e.g. [16]) use epidemic routing as an *upper bound* for evaluating their performance; i.e. they always perform worse than epidemic geocasting.

II. Challenges in Geocasting Messages in Opportunistic Networks

Before proceeding with the detailed description of the proposed geocasting protocol, we briefly discuss challenges that are relevant to geocasting in opportunistic networks; challenges that influenced our work. Geocasting in opportunistic, delay-tolerant networks is a challenging task that entails both spatial and temporal aspects and needs to take into account constraints with respect to the usage of network and device resources.

In opportunistic networks, all devices are mobile (e.g. smartphones and tablets), they move around freely carried by their users and do not have any information about the topology of the network. End-to-end paths among mobile devices do not exist and connectivity is intermittent. Devices can only discover neighbouring nodes in their vicinity and send/receive information through one of their short-range wireless interfaces (e.g. Bluetooth, Wi-Fi Direct). The network is, in principle, infrastructure-less although devices could be connected to a cellular or Wi-Fi network. Devices support location services, which may vary in the supported accuracy (and the associated battery consumption). Access to GPS for outdoors scenarios is ideal, although in most cases, coarser-grained estimations are fine. For indoors scenarios, respective localisation approaches [18] can be used.

In geocasting, the goal is to successfully deliver a message to all users (or to as many as possible) inside a specific geographical area within a specified time interval; i.e. it is not only necessary for a message to reach a cast, but it must also be efficiently disseminated within the cast. The temporal aspect is important because in many communication scenarios, messages should be invalidated and deleted from the network, either because the information they carry expires or just because the network cannot cache a message forever. The protocol should narrow down the potential recipients by defining a delivery time interval, and only the nodes that are present in the cast within that *particular time interval* should *receive* the message.

Messages in geocasting are destined to a specific location, therefore some kind of destination information must be included in the message, as the *Endpoint Identifier (EID)*. For example, if casts are pre-defined at deployment time and known to all devices, a message may carry a cast identifier; otherwise, the cast definition (e.g. centre/radius pair or coordinates of a polygon, as in our approach) must be in the message. Whenever a node receives a message, it compares

its own location with the EID of the message. This device is a recipient of the message if it currently resides within the cast defined by the EID (and the message has not expired yet).

Opportunistic networks employ *store-carry-forward* based mechanisms for message routing (including geocasting), therefore mobile devices must be able to temporarily store and carry messages before they forward them to other devices. Conceptually, the network can be seen as a buffer of finite size, which is made of all available device buffers, collectively. Although devices' memory has grown over the past years, one would not expect to be able to utilise more than some tens of MBs of memory in each device, given that other applications and background services require access to ever increasing chunks of memory. This has implications in the way data is forwarded. For example, unconstrained message flooding would result to quickly filling up devices' buffers, resulting in the quick extinction of a large number of messages[5].

Increasing the size of available buffers in each device does not simply solve the problem described above. Network bandwidth is limited but most importantly the time period that two devices can exchange messages is short given that users move. As a result, if very large buffers were used, only a very small portion of the carried messages could be forwarded from device to device[6].

Forwarding messages also comes with a cost in terms of battery consumption. Control messages exchanged among devices (e.g. to build mobility maps [15]) as well as local computations of metrics that influence how routing is done (e.g. in [16]) may result in quick draining of the device batteries. Exchanging location-related information among devices also has privacy implications that must be taken into consideration. It is worth pointing out that the network density in terms of connected mobile devices may vary significantly in different opportunistic communication scenarios. For instance, flooding the network may work well in very sparse scenarios, although the network overhead would be significantly increased as the number of users increases. Accordingly, a protocol that forwards packets very selectively (e.g. by calculating cast visiting probabilities [16]) may result in low network overhead in dense scenarios but very low message delivery ratio in sparse scenarios. In any case, a geocasting protocol should be as less sensitive to network density as possible.

III. Design

In this section we describe GSAF in details. GSAF's design has been influenced by the challenges identified in Section II.

A. Cast Definition and Membership Check

Geographical casts effectively define a group of users that reside in the same region and can be addressed by geocasting messages to this specific cast. In the following, we describe

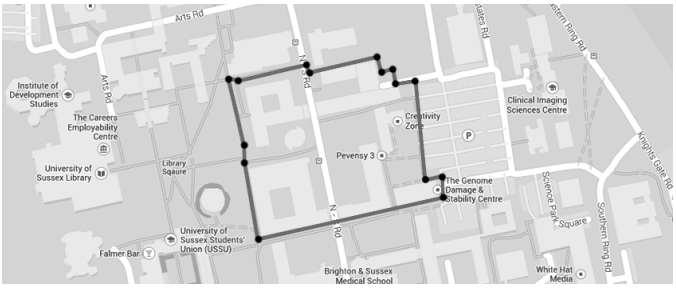


Fig. 1: Cast definition on a map

(1) how casts are defined[7] and (2) how mobile devices check whether they are recipients of a message or not.

A cast is defined as a set of coordinates in a two-dimensional space (the network). The coordinates define the edges of the cast. Figure 1 depicts an example of a cast definition inside a map. With this approach users can send their messages to arbitrarily defined casts. This provides great flexibility, potentially minimising the number of devices that are receiving unwanted messages, compared to other geocasting approaches that define casts as circles (i.e. as centre/radius pairs). With the mentioned approaches, if a specific region, which is far from the centre of the circle, needs to be reached, the radius has to be increased resulting in wasted network bandwidth for messages that reach devices for which the message is useless. In our approach, users can draw casts on their mobile phones where their messages will be destined. Messages carry the defined cast information (the set of coordinates) as their delivery address.

When a mobile device receives a message destined to a cast, it needs to identify whether it is located inside or outside the cast; i.e. whether the device is a recipient or not. Given that this check is performed for every received message, the algorithm must be very fast. Indeed, a number of very efficient algorithms have been proposed in the past in the context of the *Point in Polygon* problem, a well studied problem in computer graphics and image processing [19]. As its name suggests, solutions to this problem check whether a specific point is inside a polygon or not. According to Haines [19], three main techniques can be used to solve this problem; the *crossing test*, *angle summation test* and *triangle test*. Among these, the crossing test is the fastest (as shown in [19]) and therefore has been adopted in our work. An example of the crossing test is illustrated in Figure 2. Initially, a vertical (to the x-axis) line that crosses the point (with coordinates (x_p,y_p)) that needs to be checked is drawn. The point (x_p,y_p) is the *initial point* that decomposes this line into two *rays* (half-lines). The number of intersections of one of the rays (e.g. the solid line in Figure 2) with the sides of the polygon is used to check whether the point is in the polygon or not; if the number is

[5]Indeed, this is clearly shown in Section IV where the delivery ratio for epidemic geocasting is low.

[6]This is also shown in our evaluation section, where the delivery ratio does not increase as the available buffers size does.

[7]Note that casts not needed to be pre-defined. Instead, a sender can define a cast to send a message to on-the-fly.

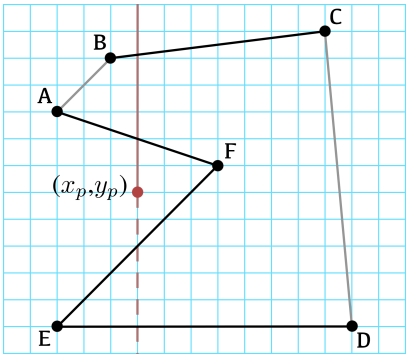


Fig. 2: A crossing test example

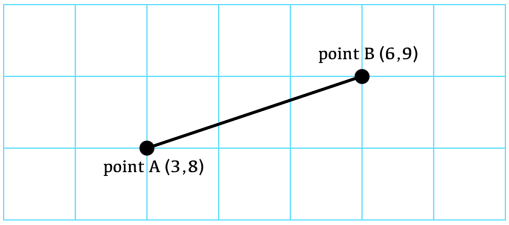
even, the point is located outside the polygon, otherwise the point is inside.

For each pair of neighbouring polygon coordinates, we calculate the parameters of the line equation that defines the line that connects these two points, as shown in Figure 3. The x_{range} defines the projection of each side of the polygon to the x-axis. In order to calculate the intersections of the vertical line with the given polygon, we simply test whether x_p is within the x_{range} for each side of the polygon. For example, in Figure 2, sides *BC*, *DE*, *EF* and *FA* intersect with the vertical line. Next, we calculate the y coordinates of the intersection points by solving the line equation that defines each side of the polygon using x_p. Finally, we count the number of y coordinates that are larger than y_p (i.e. looking at the *ray* illustrated with solid line). Based on this number we decide whether the device (x_p,y_p) is inside (odd number) or outside (even number) the cast (the polygon). In our simulation model, the crossing test implementation performs slightly worse compared to just checking whether a point is within a circle. We argue though that this slight performance penalty is negligible with respect to routing performance and battery consumption.

B. GSAF Routing Protocol

Messages in our approach include the cast definition (as a set of two-dimensional coordinates), a pair of epoch times that define the time period during which the message is valid, a number of *tickets* (represented as a value *T*) (see Section III-B1) and the actual payload. Note that in our approach messages can become valid after their creation and initial forwarding. One could therefore account for the cast delivery latency and make the messages valid in the near future, when it is anticipated that they will reach the destined cast. This can also come in handy in scenarios where messages are created a priori. For example, in a geographically targeted advertisement scenario one could prepare several messages that can become valid at specific times during the day. Mobile devices are assumed to be loosely synchronised, a fair assumption for today's smartphones.

GSAF gets inspiration from well-established DTN routing protocols, such as *spray-and-wait* [17] and *epidemic* routing [20]. These are unicast protocols and use EIDs to identify



Linear equation :

$$y - y_A = \frac{y_B - y_A}{x_B - x_A}(x - x_A) \quad x_{range} : [\, x_A, x_B \,] \Rightarrow$$
$$y - 8 = \frac{9-8}{6-3}(x-3) \quad x_{range} : [\, 3, 6 \,] \Rightarrow$$
$$y = -\frac{2}{3}x + 8 \quad x_{range} : [\, 3, 6 \,]$$

Fig. 3: Calculating the parameters of the line equation

individual devices. Geocasting, instead of unicasting messages, requires adapting EIDs so that they identify casts. GSAF consists of two phases: (1) The message is forwarded to the destination cast and (2) it is flooded to all nodes in the cast, in a controlled fashion where messages are not allowed to exit the cast.

1) Phase 1 - Forwarding (and carrying) the message to the destination cast: The first phase follows a multi-copy spraying approach (inspired by [17]), which is fast in terms of reaching the destination cast as well as efficient in terms of message delivery and network overhead. Upon creation, a number of tickets *T* is "assigned" to the message (represented as a ticket counter which is included in the message). *T* denotes the number of times a carried message can be replicated to encountered devices. Each time a message is copied and forwarded to another node, *T* is decreased by 1 in both messages. When *T* gets to zero, the message cannot be forwarded any further; i.e. it will be deleted when the local buffer gets full or when the message expires.

2) Phase 2 - Delivering the message to all nodes inside the cast: In the second phase, the message is disseminated to all nodes inside the cast by following an intelligent flooding approach. Our protocol floods the message to nodes inside the cast by handing a copy to them. If a copy of the message goes out of the destination cast, it can only be forwarded back to nodes inside the cast (this is enforced by line 20 in Algorithm 1 where *T* is set to 0 at the beginning of the second phase). The message will sit in the device's buffer until it expires. At that point it will be deleted. This way we prevent unnecessary message exchanges outside the cast, which, given their flooding nature, would increase the network overhead significantly and could fill up the buffers of nearby devices quickly.

Both phases are concisely described in Algorithm 1. Figure 4 illustrates an example of how messages are disseminated in our approach. The sender (shown in green) creates a new message and initialises *T* to 3. It encounters two nodes (the one after the other) and for each such encounter, it decreases the value of *T* in the message and forwards a copy of the message to the remote node. As shown in the figure, *T* is first decreased

to 2 (which is also the value of T in the message received by the node above the source node) and, then, to 1 (the value of T in the message received by the node below the source node). The same takes place when these two nodes encounter other nodes in the opportunistic network. At the end of the illustrated example, a number of nodes roam outside the cast carrying a message with a T value of zero. These nodes will not forward the message any further. One node that resides inside the cast has also received the message (T is zero) but the message will keep being forwarded to recipients inside the cast, as phase 2 suggests. More generally, a message can end up inside the destination cast either after it was exchanged between a node outside and a node inside the cast or because it was physically carried by a node to inside the cast. In both cases, T can have any value (equal or greater than 0). T will be set to 0 at the beginning of the second phase.

The value of T can be pre-specified for specific network deployments (e.g. for communication within a city) based on e.g. the expected node density and mobility patterns. In Section IV we present a sensitivity analysis for the initial value of T, which indicates that values close to the optimal one (with respect to message delivery ratio and latency), also result to very good performance. One could therefore dynamically set *T's* initial value e.g. by estimating the density of mobile devices, as in [21]. Research on building mobility maps [15] could also complement our work.

C. Buffer scheduling policy

The algorithm presented in Algorithm 1 involves the execution of a buffer scheduling policy (at line 4), which defines the order in which stored messages will be exchanged. Devices store messages in their buffers, carry and forward them to other devices in the network. Given that network bandwidth is finite and the available time that two devices can communicate as they move is limited, a device will have to prioritise message exchanges. In the context of our work, we have investigated the usage of the following policies:

- First In First Out (FIFO): Messages are forwarded to remote devices in the order they have been received.
- Last In First Out (LIFO): Messages are forwarded to remote devices in the reverse order compared to the one they were received.
- Highest TTL First Out (HTFO): The message with the longest lifetime is forwarded first.
- Lowest TTL First Out (LTFO): The message with the shortest lifetime is forwarded first.

In Section IV-C, we extensively evaluate the policies above with respect to the main network metrics investigated in this paper (see Section IV-A).

Algorithm 1 GSAF Router

```
1:  for each encounter between two hosts in the network do
2:      if the host is the sender then
3:          drop expired messages from buffer
4:          apply buffer scheduling policy
5:          for each message in the buffer do
6:              if (T > 0) then
7:                  subtract T by 1
8:                  forward a copy to the receiver
9:              else if (T = 0) then
10:                 if sender is inside the destination cast then
11:                     forward a copy to the receiver
12:                 end if
13:             end if
14:         end for
15:     else if the host is receiver then
16:         for each received message do
17:             if device is in destination cast then
18:                 deliver to user
19:                 store a copy into buffer
20:                 set T equal to 0
21:             else if device outside the destination cast then
22:                 store message into buffer
23:                 subtract T by 1
24:             end if
25:         end for
26:     end if
27: end for
```

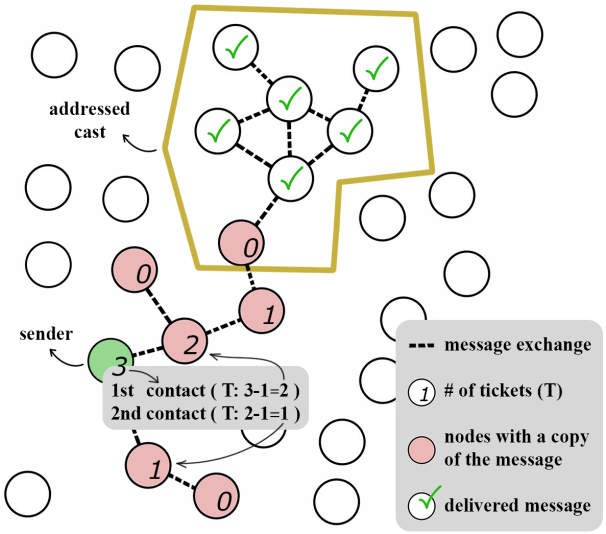


Fig. 4: GSAF Routing scheme

IV. Evaluation

In this section we present an extensive experimental evaluation of our geocasting protocol. We have implemented GSAF in the *One* simulator [22]. We chose *One* because it supports (1) several realistic mobility models, (2) an extensible architecture for implementing routing protocols and sender/receiver types and (3) visualisation of both node mobility and message exchanging in real time [23].

A. Metrics and Measurements

We evaluate the performance of our protocol through two metrics, namely the *message delivery ratio* and *message delivery latency*.

Message delivery ratio. In geocasting, messages are not addressed to specific devices but to geographical areas where multiple devices may reside during the lifetime of a message. In contrast to unicast protocols where the delivery status of a message can have two values (delivered or undelivered),

in geocasting one has to take into account both the spatial and temporal aspects. The message delivery ratio for a unicast protocol would be calculated as follows:

$$unicast\ delivery\ ratio = \frac{number\ of\ delivered\ messages}{number\ of\ created\ messages} \quad (1)$$

In geocasting, each message has a delivery ratio itself, instead of a mere delivery status (delivered or undelivered). The per-message delivery ratio (*pmdr*), which is the fraction of the number of devices that were located in the cast throughout the lifetime of the message and received the message (*actual number of recipients*) to the total number of devices that should have received the message (*total number of recipients*), is calculated as follows:

$$pmdr = \frac{actual\ number\ of\ recipients}{total\ number\ of\ recipients} \quad (2)$$

Equation 2 calculates the delivery ratio of a single message. The overall delivery ratio (odr) of the geocasting protocol is measured based on the delivery ratio of all created messages, as shown in Equation 3.

$$odr = \frac{pmdr\ 1\ +\ pmdr\ 2\ +\ \cdots\ +\ pmdr\ n}{number\ of\ created\ messages} \quad (3)$$

Message delivery latency. The same rationale is followed when measuring the message delivery latency. We measure the time it takes for a message to reach a destination device (i.e. a device in the destination cast) and calculate the per-message delivery latency. The overall delivery latency is the average for all created messages.

B. Simulation scenarios

We simulated the operation of GSAF in Helsinki city centre (an area of $4500m \times 3400m$)[8]. Although GSAF does not rely on static, pre-defined casts, for evaluation purposes only, we have created 16 casts, as illustrated in Figure 5, and messages were destined . In our simulations, we experimented with 8 different levels of user density (65, 130, 195, 260, 325, 390, 455 and 520). All simulations involve three types of users: Pedestrians, Cars and Buses. The detailed amount of users for each model is shown in Table I.

The movement models of all three simulated types of users are shown in Table II. The default total number of users in our simulations, unless otherwise mentioned, is 195. The respective numbers for each type is shown in Table I.

In addition, we have experimented with different sizes of device buffers (5, 10, 15, 20, 25 and 30 MB) (the default being 10 MB) and different message lifetimes (30, 60, 90, 120, 150, 180, 210 and 240 minutes) (the default being 120 minutes). We have simulated our protocol with two wireless interfaces; (1) Wi-Fi 802.11ac with transmission speed of 433 Mbps and range of 20 meters. and (2) Bluetooth 802.16 v4.0

[8]We also simulated our protocol in a custom map of the University of Sussex campus. The results were very consistent with what is presented in this paper. Due to lack of space, we do not include any graphs but the interested reader can read more at *http://www.aydinrajaei.com/research/gsaf-project/*.

Fig. 5: Pre-defined casts in the city centre of Helsinki

TABLE I: Number of hosts for different user density levels

Total #	**65**	**130**	**195**	**260**	**325**	**390**	**455**	**520**
Pedestrian	40	80	120	160	200	240	280	320
Car	20	40	60	80	100	120	140	160
Bus	5	10	15	20	25	30	35	40

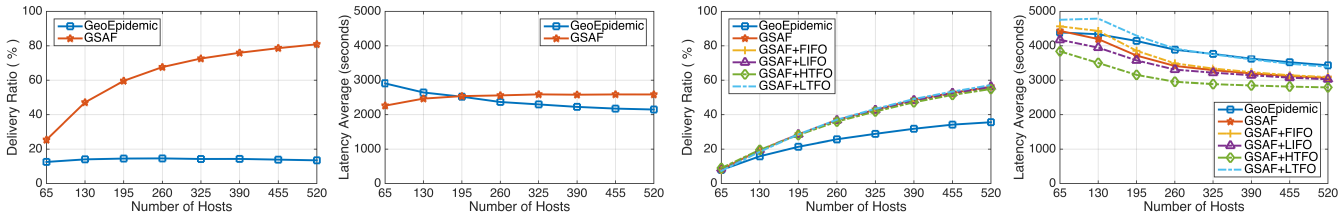
TABLE II: Users' movement models

	Speed	**Movement Model**
Pedestrian	0.5 - 1.5 (m/s)	Shortest Path Map Based Movement
Car	2.7 - 13.9 (m/s)	Shortest Path Map Based Movement
Bus	7 - 10 (m/s)	Bus Movement

with transmission speed of 2 Mbps and range of 10 meters. All simulations ran for 16 hours; the warm-up and cool-down periods for each simulation was 2 hours (therefore our results are drawn from 12 hours of simulated time for each run). We repeated every simulation 5 times with different seeds for the mobility models. We schedule messages as follows: a sender and a destination cast are selected uniformly at random from the set of devices in the networks and the set of pre-defined casts, respectively; message payload is fixed (500KB) and a new message is scheduled every 25 to 35 seconds (a values selected uniformly at random from this time range). Unless otherwise stated, the buffer scheduling policy is random, which means that messages are randomly selected from the device buffer when a device encounters another device in the network. We also experiment with buffer scheduling policies defined in Section III-C (i.e. FIFO, LIFO, HTFO and LTFO). Finally, we compare our geocasting protocol with *GeoEpidemic*, an adaptation of the epidemic protocol [20] for geocasting. Note that GeoEpidemic is used as an upper performance bound in other papers [16]. In our work, GeoEpidemic gives us a lower bound, which means that our protocol always performs better.

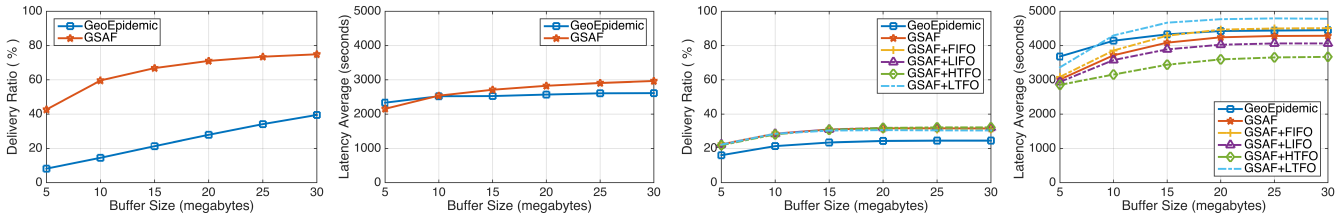
C. Evaluation results

In this section we discuss the results of the experimental evaluation. In total, we collected data from 1420 different simulations. The total simulated time has been 22,720 hours. The actual time it took to execute all simulations has been around 18,000 hours. We investigate the influence of (1) user density, (2) available buffers size and (3) message lifetime in



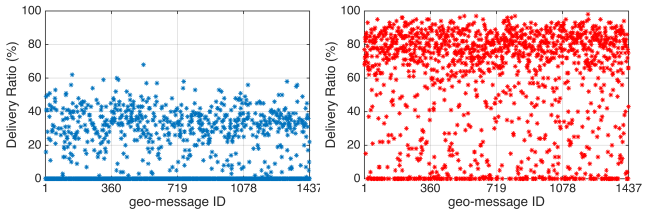
(a) Delivery Ratio (Wi-Fi) (b) Delivery Latency (Wi-Fi) (c) Delivery Ratio (Bluetooth) (d) Delivery Latency (Bluetooth)

Fig. 6: Influence of user density on geocasting performance



(a) Delivery Ratio (Wi-Fi) (b) Delivery Latency (WiFi) (c) Delivery Ratio (Bluetooth) (d) Delivery Latency (Bluetooth)

Fig. 7: Influence of buffer capacity on geocasting performance



(a) GeoEpidemic (b) GSAF

Fig. 8: Per-message delivery ratio

the performance of the proposed protocol (see performance metrics described in Section IV-A).

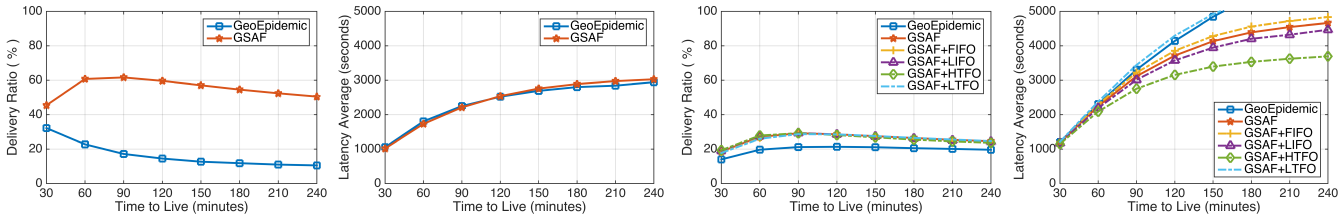
Note that in the sections below, we do not present the results for different buffer scheduling policies when devices communicate through Wi-Fi. This is because the buffer scheduling policy has no effect when a device can transfer *all* its stored messages to a remote device when they encounter each other; and this is the case when Wi-Fi is used to exchange buffers up to 30MB (as in our simulations), for all types of users. However, when communication takes place through Bluetooth, the adopted buffer scheduling policy plays a crucial role in the performance of the gocasting protocol.

1) Influence of user density: For this set of simulations, we used the default values for the buffer size and message lifetime and we varied the number of users in the map. Figure 6 shows the results. We observe that the increase in user density improves the message delivery ratio. The average latency is not significantly affected, although there is a an improvement in the Bluetooth case. GSAF delivers a higher ratio of message compared to GeoEpidemic, while maintaining similar levels of delivery latency (although in the Bluetooth scenario GSAF performs much better). We highlight that, as explained in Section IV-C4, our protocol, not only outperforms GeoEpidemic, but also does so with an extremely lower network overhead (in terms of exchanged messages) compared to GeoEpidemic. Also note that the observed delivery ratio for GeoEpidemic decreases when the supported transmission rate is higher (i.e. in the Wi-Fi case). This counter-intuitive observation is because when more messages are exchanged in a *flooding* fashion, then the probability that a message will disappear from all buffers very quickly increases.

As mentioned above, buffer scheduling policies only have an effect on performance when it is not possible for a device to exchange its entire buffer upon encounter with another device. The policy prioritises the way messages are exchanged (see Section III-C). As seen in Figure 6 (c and d), the delivery ratio is insensitive to the followed scheduling policy. However, they affect the message delivery latency. HTFO performs the best. The reason for that is because with HTFO messages that have the higher chance to reach the cast (given their large lifetime) are *consistently* prioritised over messages that are likely to not make it to the cast (and that is why the LTFO policy performs the worst).

In order to grasp a better idea about how values of message delivery ratio are distributed for all different messages, we present Figure 8, which depicts a scatter plot for per-message delivery ratios (for 1437 messages) for both GSAF and GeoEpidemic. These results are extracted by running a simulation with the default values, as described in Section IV-B. According to the figure, the accumulation of message ratios in our protocol is above the average delivery ratio ($\sim$60%). As mentioned above, GeoEpidemic performs much worse. Note that the number of messages that are never delivered to the destination cast is $\sim$150 for GSAF and $\sim$800 for GeoEpidemic.

2) Influence of buffer capacity: For the second experiment, the user density and message lifetime remain unchanged in order to observe the impact of the buffer size on the performance of the proposed protocol. The buffer is able to keep certain amount of messages based on its overall capacity. As discussed at the beginning of this paper, DTNs use *store-carry-forward* scheme to exchange information between



(a) Delivery Ratio (WiFi) (b) Delivery Latency (WiFi) (c) Delivery Ratio (Bluetooth) (d) Delivery Latency (Bluetooth)

Fig. 9: Influence of message lifetime on geocasting performance

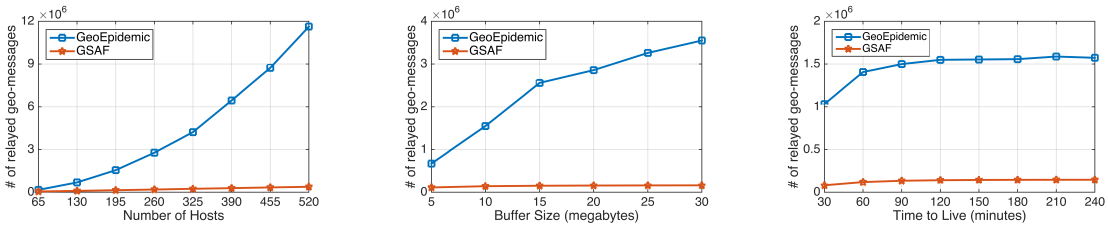


Fig. 10: Number of relayed message copies in the network (less is better)

the hosts. The greater the buffer size, the more messages a device can carry around to deliver to the destination cast. The problem with large buffers, apart from the fact that they are rather expensive, is that in opportunistic networks, there is not enough time to exchange all buffered messages, given the limited bandwidth and, more importantly, the mobility of users. Figure 7 confirms the fact that GSAF performs better compared to GeoEpidemic. When Wi-Fi is used, the message delivery ratio increases with the buffer availability because a device can exchange all its messages with other devices when they encounter each other. However, the situation is very different when Bluetooth is used; the delivery ratio reaches a plateau (∼30%) when the buffer size becomes 15 MB.

The results for the delivery latency follow a similar pattern compared to the previous set of simulations. Note that the latency increases along with the buffer size (for all buffer scheduling policies) for the reasons mentioned above. In this case, the buffer capacity of 20 to 25 MB seems enough for GSAF to handle the network on its best. Overall, our protocol is more efficient with respect to buffer usage compared to GeoEpidemic.

3) Influence of message lifetime: The third experiment is designed to study the impact of message lifetime on the overall performance of the geocasting protocol. As discussed in the design section, we define a delivery interval during which a message is valid. This interval effectively narrows down the amount of recipients located in a geographical cast. As shown in Figure 9, the delivery ratio for both GSAF and GeoEpidemic decreases as the message lifetime increases. This is counter-intuitive and highlights the rather complex nature of opportunistic networks; one would expect that as the lifetime of a message increases, then there would be more time to deliver it in its destination cast. However, longer lifetimes imply the need for larger buffers to store (and carry) the messages and higher bandwidth to exchange them. Given the finite (and rather limited) nature of both of them the delivery ratio actually decreases as the message lifetime increases because messages are being extinct due to the lack of buffer availability. Also note that longer message lifetimes mean larger number of recipients (that resided in the cast within the message lifetime) which through time they may have moved out of the cast.

The results for the delivery latency follow a similar pattern with the ones presented above. The HTFO buffer scheduling policy performs the best for the same reasons as the ones described in Section IV-C1.

4) Network overhead: As mentioned in Section II, a routing protocol (including a geocasting one) for opportunistic networks must keep the number of exchanged messages low, while still being able to perform well. Exchanging more messages means higher battery consumption and requires more bandwidth and larger buffers. In the following we investigate the network overhead, in terms of the total number of relayed copies of messages, by looking at the results of the simulation with the default parameters. Figure 10 presents the total number of relayed message copies for different numbers of users, buffer sizes and message lifetimes, respectively. It is obvious that the GeoEpidemic protocol, which follows a flooding approach, attempts to send as many messages as possible (e.g. as the number of devices in the network increase or as the available buffer size increases). In contrast, GSAF always limits the number of messages by employing the ticket-based approach, as explained in Section III-B. Our protocol manages to keep down the number of relayed messages to around 150,000 copies, while GeoEpidemic sends up to 12 million copies.

5) Influence of the number of initial tickets (T): GSAF's first phase heavily relies on the number of tickets (T) a message is assigned with upon its creation. As mentioned in Section III-B, the value of T could be dynamically adjusted (e.g. based on the inferred device density) to a value that provides the best performance. In this section we investigate how different values of T influence GSAF's performance. We are interested in looking at how sensitive GSAF is to

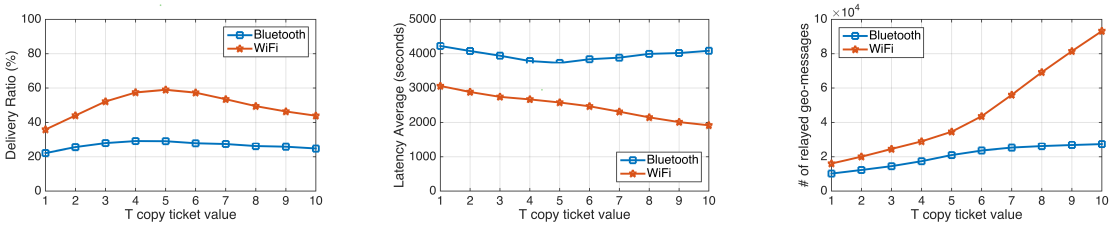


Fig. 11: Sensitivity analysis for T

T, which, in turn, means what the penalty of misconfiguring T in a dynamic approach would be. The results shown in Figure 11 are promising. The delivery ratio is not sensitive to T and, therefore, values around the optimal one (for the specific default simulation scenario) would perform adequately well (i.e. ∼60% compared to ∼53 - 58%). The results are similar for the message delivery latency. Finally, it is well-expected that the number of relayed messages will increase as T increases. Note that when Bluetooth is used, this number reaches its highest when T is 9; after that point the bottleneck becomes Bluetooth's bandwidth.

V. Conclusion

In this paper we have presented *Geocasting Spray And Flood (GSAF)*, a simple but efficient an flexible protocol for geocasting messages in opportunistic, delay-tolerant networks. We highlighted significant challenges in geocasting in the context of opportunistic networks and described how GSAF deals with these challenges, overcoming limitations of existing approaches. GSAF delivers messages to their destination casts in two phases. During the first one a message is replicated in a controlled way (using a ticketing mechanism) to encountered devices. When the message reaches its destination cast, GSAF's second phase is enabled and the message is carefully flooded within the limits of the cast. Casts need not be pre-defined and users are free to define their own casts even on a per-message basis. Casts are polygons in a two-dimensional space, allowing for flexible and efficient information dissemination.

We implemented GSAF in the *One Simulator* and extensively evaluated its performance and general behaviour using a large range of simulations. GSAF always performs better compared to GeoEpidemic, a geocasting adaptation of the epidemic routing protocol. GeoEpidemic has been used as an upper bound benchmark in other papers but for GSAF is a lower bound. GSAF is also battery- and network-friendly by requiring less relayed message copies, compared to GeoEpidemic, to reach the destination casts for each message. Finally, we presented results that indicate, the value of the initial tickets assigned to a message can be dynamically adjusted based on e.g. user density or mobility patterns, since slight misconfiguration do not severely affect GSAF's performance.

References

[1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003, pp. 27–34.

[2] K. Fall and S. Farrell, "Dtn: an architectural retrospective," *Selected Areas in Communications, IEEE Journal on*, vol. 26, no. 5, pp. 828–836, 2008.

[3] I. Wakeman, S. Naicken, J. Rimmer, D. Chalmers, and C. Fisher, "The fans united will always be connected: Building a practical DTN in a football stadium," in *ADHOCNETS*, 2013.

[4] S. Trifunovic, B. Distl, D. Schatzmann, and F. Legendre, "WiFi-Opp: Ad-hoc-less opportunistic networking," in *Proc. of ACM CHANTS*, 2011.

[5] M. Grossglauser and D. Tse, "Mobility increases the capacity of ad hoc wireless networks," *IEEE/ACM Transactions on Networking*, 2002.

[6] V. Vukadinović and G. Karlsson, "Spectral efficiency of mobility-assisted podcasting in cellular networks," in *MobiOpp*, 2010.

[7] J. Ott, E. Hyytia, P. Lassila, T. Vaegs, and J. Kangasharju, "Floating content: Information sharing in urban areas," in *PerCom*, 2011.

[8] B. Han, P. Hui, V. A. Kumar, M. V. Marathe, G. Pei, and A. Srinivasan, "Cellular traffic offloading through opportunistic communications: A case study," in *ACM CHANTS*, 2010.

[9] H. Luo, R. Ramjee, P. Sinha, L. E. Li, and S. Lu, "UCAN: A unified cellular and ad-hoc network architecture," in *MobiCom*, 2003.

[10] J. Erman and K. Ramakrishnan, "Understanding the super-sized traffic of the super bowl," in *IMC*, 2013.

[11] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang, "A first look at cellular network performance during crowded events," in *SIGMETRICS*, 2013.

[12] Y. Cao and Z. Sun, "Routing in delay/disruption tolerant networks: A taxonomy, survey and challenges," *Communications Surveys & Tutorials, IEEE*, vol. 15, no. 2, pp. 654–677, 2013.

[13] C. Maihofer, "A survey of geocast routing protocols," *Commun. Surveys Tuts.*, vol. 6, no. 2, pp. 32–42, Apr. 2004. [Online]. Available: http://dx.doi.org/10.1109/COMST.2004.5342238

[14] Y. Ma and A. Jamalipour, "Opportunistic geocast in disruption-tolerant networks," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec 2011, pp. 1–5.

[15] M. Piorkowski, "Mobility-centric geocasting for mobile partitioned networks," in *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, Oct 2008, pp. 228–237.

[16] S. Lu and Y. Liu, "Geoopp: Geocasting for opportunistic networks," in *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, April 2014, pp. 2582–2587.

[17] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: an efficient routing scheme for intermittently connected mobile networks," in *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*. ACM, 2005, pp. 252–259.

[18] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," *Communications Surveys Tutorials, IEEE*, vol. 15, no. 3, pp. 1281–1293, Third 2013.

[19] E. Haines, "Point in polygon strategies," *Graphics gems IV*, vol. 994, pp. 24–26, 1994.

[20] A. Vahdat, D. Becker *et al.*, "Epidemic routing for partially connected ad hoc networks," Technical Report CS-200006, Duke University, Tech. Rep., 2000.

[21] J. Weppner and P. Lukowicz, "Bluetooth based collaborative crowd density estimation with mobile phones," in *Pervasive Computing and Communications (PerCom), 2013 IEEE International Conference on*, March 2013, pp. 193–200.

[22] A. Keränen, J. Ott, and T. Kärkkäinen, "The one simulator for dtn protocol evaluation," in *Proceedings of the 2nd international conference on simulation tools and techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2009, p. 55.

[23] The opportunistic network environment simulator. [Online]. Available: http://www.netlab.tkk.fi/tutkimus/dtn/theone/